

Xilinx sources and documentation

By Farnoud Farahmand <ffarahma@gmu.edu>

Last modified February 11, 2016

1. Board Support Package


- 1- Lunch SDK through Vivado
- 2- Create new application project
- 3- Now you have access to **system.mss** under board support package
Note: system.mss page opens automatically after creating new application project
- 4- Click on **Documentation** in front of the name of each ips which is included in your design

The screenshot shows the Vivado IDE interface. The Project Explorer on the left displays the project structure, with 'system.mss' highlighted under the 'test_3_B_bsp' directory. The main window shows the 'test_3_B_bsp Board Support Package' page. The page includes sections for 'Target Information', 'Operating System', and 'Peripheral Drivers'. The 'Peripheral Drivers' section lists various drivers, with 'axi_gpio_0 gpio', 'axi_gpio_1 gpio', and 'axi_timer_0 tmrctr' highlighted in red. Each driver name has a 'Documentation' link next to it, which is also highlighted in red. The 'SDK Log' at the bottom shows the message: '18:56:45 INFO : Launching XSDB server: xsdb.b'.

- 5- A new web page opens automatically that contains all of the information regarding the selected ip. Also, you have access to all driver functions which is available for this ip.

gpio_v4_0: Main Page

file:///C:/Xilinx/SDK/2015.4/data/embeddedsw/XilinxProcessorIPLib/drivers/gpio_v4_0/doc/html/ε



gpio_v4_0

Xilinx SDK Drivers API Documentation

Overview | Data Structures | APIs | File List

gpio_v4_0 Documentation

This file contains the software API definition of the Xilinx General Purpose I/O (XGpio) device driver. The Xilinx GPIO controller is a soft IP core designed for Xilinx FPGAs and contains the following general features:

- Support for up to 32 I/O discretes for each channel (64 bits total).
- Each of the discretes can be configured for input or output.
- Configurable support for dual channels and interrupt generation.

The driver provides interrupt management functions. Implementation of interrupt handlers is left to the user. Refer to the provided interrupt example in the examples directory for details.

This driver is intended to be RTOS and processor independent. Any needs for dynamic memory management, threads or thread mutual exclusion, virtual memory, or cache control must be satisfied by the layer above this driver.

Initialization & Configuration

The **XGpio_Config** structure is used by the driver to configure itself. This configuration structure is typically created by the tool-chain based on HW build properties.

To support multiple runtime loading and initialization strategies employed by various operating systems, the driver instance can be initialized in one of the following ways:

- XGpio_Initialize(InstancePtr, Deviceld) - The driver looks up its own configuration structure created by the tool-chain based on an ID provided by the tool-chain.
- XGpio_CfgInitialize(InstancePtr, CfgPtr, EffectiveAddr) - Uses a configuration structure provided by the caller. If running in a system with address translation, the provided virtual memory base address replaces the physical address present in the configuration structure.


Note

This API utilizes 32 bit I/O to the GPIO registers. With less than 32 bits, the unused bits from registers are read as zero and written as don't cares.

MODIFICATION HISTORY:

gpio_v4_0: Main Page | gpio_v4_0: XGpio_Config

file:///C:/Xilinx/SDK/2015.4/data/embeddedsw/XilinxProcessorIPLib/drivers/gpio_v4_0/doc/html/ε



gpio_v4_0

Xilinx SDK Drivers API Documentation

Overview | **Data Structures** | APIs | File List

XGpio_Config Struct Reference

Gpio_v4_0



gpio_v4_0

Xilinx SDK Drivers API Documentation

Overview	Data Structures	APIs	File List
-----------------	-----------------	------	-----------

Gpio_v4_0

Data Structures

struct	XGpio_Config
struct	XGpio

Macros

#define	XGpio_WriteReg (BaseAddress, RegOffset, Data) XGpio_Out32((BaseAddress) + (RegOffset), (u32)(Data))
#define	XGpio_ReadReg (BaseAddress, RegOffset) XGpio_In32((BaseAddress) + (RegOffset))

Functions

int	XGpio_CfgInitialize (XGpio *InstancePtr, XGpio_Config *Config, u32 EffectiveAddr)
void	XGpio_SetDataDirection (XGpio *InstancePtr, unsigned Channel, u32 DirectionMask)
u32	XGpio_GetDataDirection (XGpio *InstancePtr, unsigned Channel)
u32	XGpio_DiscreteRead (XGpio *InstancePtr, unsigned Channel)
void	XGpio_DiscreteWrite (XGpio *InstancePtr, unsigned Channel, u32 Data)
int	XGpio_Initialize (XGpio *InstancePtr, u16 Deviceld)
XGpio_Config *	XGpio_LookupConfig (u16 Deviceld)
void	XGpio_DiscreteSet (XGpio *InstancePtr, unsigned Channel, u32 Mask)
void	XGpio_DiscreteClear (XGpio *InstancePtr, unsigned Channel, u32 Mask)
int	XGpio_SelfTest (XGpio *InstancePtr)
void	XGpio_InterruptGlobalEnable (XGpio *InstancePtr)
void	XGpio_InterruptGlobalDisable (XGpio *InstancePtr)
void	XGpio_InterruptEnable (XGpio *InstancePtr, u32 Mask)
void	XGpio_InterruptDisable (XGpio *InstancePtr, u32 Mask)
void	XGpio_InterruptClear (XGpio *InstancePtr, u32 Mask)
u32	XGpio_InterruptGetEnabled (XGpio *InstancePtr)
u32	XGpio_InterruptGetStatus (XGpio *InstancePtr)

- 6- If you click on each of these function names you will be directed to complete instruction for using these functions.

```
void XGpio_SetDataDirection ( XGpio * InstancePtr,
                             unsigned Channel,
                             u32 DirectionMask
                             )
```

```
#include <xgpio.c>
```

Set the input/output direction of all discrete signals for the specified GPIO channel.

Parameters

- InstancePtr** is a pointer to an **XGpio** instance to be worked on.
- Channel** contains the channel of the GPIO (1 or 2) to operate on.
- DirectionMask** is a bitmask specifying which discretions are input and which are output. Bits set to 0 are output and bits set to 1 are input.

Returns

None.

Note

The hardware must be built for dual channels if this function is used with any channel other than 1. If it is not, this function will assert.

References **XGPIO_TRI_OFFSET**, and **XGpio_WriteReg**.

2. DocNav application

This application is installed during Xilinx Vivado installation Process. If you search the name of each ip in this application, you will access a separate and complete user manual of each ip.

The screenshot shows the Xilinx Documentation Navigator 2015.4 - Catalog View. The interface includes a search bar at the top with the text "Find in Grid: AXI GPIO" and a search icon. Below the search bar, there are several document filters on the left side, including "Silicon Devices", "Design Tools", "ISE Design Suite", "IP", and "Document Types". The main area displays a list of documents, with "LogiCORE IP AXI GPIO (v1.01b) Data Sheet (AXI)" highlighted in blue. The search results are displayed in a list view, and the number of documents displayed is 2981 out of 3725.